Sensors and Materials, Vol. 31, No. 6 (2019) 2143–2154
MYU Tokyo

S & M 1920

Analysis and Forecasting for Traffic Flow Data

Yitian Wang[1,2*] and Joseph Jaja[2]

[1]Shenyang University of Chemical Technology,
School of Information Engineering, Shenyang, Liaoning 110142, China
[2]University of Maryland – College Park,
Department of Electric and Computer Engineering, College Park, MD 20740, USA

(Received January 11, 2019; accepted April 25, 2019)

Keywords: pattern discovery, unsupervised machine learning, principal component analysis (PCA), short-term real-time forecasting, intelligent transportation

The urban transportation system involves the challenging task of transferring people and materials across densely populated areas, and hence its operational efficiency directly affects the entire city. In this study, we overcome the restriction of both time and space by introducing an online version of the principal component analysis (PCA), called the projection approximation subspace tracking with deflation (PASTd) algorithm. The algorithm is implemented to derive core traffic patterns of traffic flow data of Baltimore, Maryland, US. The *k*-nearest-neighbor (KNN) method is applied to predict the values of these core traffic patterns in the near future. Thus, the traffic information of Baltimore County can be forecasted with linear complexity and traffic congestion can be traced with little latency. Unlike traditional traffic prediction methods, our method aims at network-level prediction, regardless of urban or freeway road segments. The results show that our forecasting method is efficient, flexible, and robust.

1. Introduction

Traffic data information plays a very important role in our daily life. As a result, considerable work has been done for generating and analyzing traffic time series.[1–3] In general, we regard traffic data as the streaming data generated at regular time intervals. At each time step, we receive traffic information about a large number of road segments, which has to be analyzed and disseminated in real time. On the other hand, vehicle operators would like to receive immediate up-to-date traffic summaries and cannot afford any postprocessing.[4]

An effective way to handle this problem is to reduce the large volumes of traffic data into a small number of meaningful trends that can be updated and broadcasted in real time. Traffic flow data correlate with the information about most road segments. Hence, one possible research direction is to use clustering algorithms to group together road segments that follow similar traffic patterns. Instead of analyzing the traffic of n road segments, we can analyze the patterns of k groups, where k is much smaller than n. Applying self-organizing maps (SOMs) is a possible approach, which has been used in postprocessing analysis.[5] Mixture models present good opportunities for streaming data analysis[6,7] as well. Here, we pursue a different approach

*Corresponding author: e-mail: wytq0628@163.com
https://doi.org/10.18494/SAM.2019.2315

ISSN 0914-4935
https://myukk.org/

based on finding patterns (or hidden variables) such that the time series of average speeds for each road segment can be generated using a linear combination of these patterns. References 8–10 explicitly focus on discovering hidden variables. In CluStream,[8] patterns are found by an offline strategy based on stored data. Sakurai *et al.*[9] determined lag correlations among multiple streams. StatStream[10] uses discrete Fourier transform (DFT) to summarize streams within a finite window size. We would like to find a method that can find a relatively few inherent patterns in an online fashion with linear complexity, with no need for data buffering.

Short-term traffic prediction plays a crucial role in an intelligent transportation system (ITS). With reliable forecasting data, administrators can manage traffic networks effectively and travelers can decide on departure time or travel routes more easily.[11] Many statistical models have been proposed for short-term traffic forecasting. For example, time series models,[12,13] Bayesian models,[14] Kalman filter models,[15] and support vector machine regression models[16] have been widely applied to predict motorway and freeway traffic conditions. Neural network models using artificial intelligence algorithms[17–19] and unsupervised machine learning algorithms[20] have also gained researchers' attention recently. Until now, most models focus on motorways and freeways.[21,22] A network-level method is needed for better prediction. We require that the prediction method is efficient and scalable. Even though the number of road segments can become very large, the method should be able to make reliable prediction in real time. In this paper, we propose a method that can meet all the following requirements: online use, linear complexity, no need for data buffering, scalability, network level, and reliability.

2. Pattern Discovery for Traffic Flow Data

Problem Formulation Given n time series corresponding to average speeds on n road segments, updated at each time step t, we aim to determine k hidden variables, where $k \ll n$, such that the linear combinations of these k hidden variables can be used to reconstruct the time series data. Thus, the dimension of the data set is significantly reduced. As a result, we can make more effective, low-cost prediction for speeds in the near future. Figure 1 shows an example of a time series of average speeds for a road segment over a week. The x-axis represents the minutes in a week ranging from 1 to 10080, while the y-axis represents the corresponding average speed in mph.

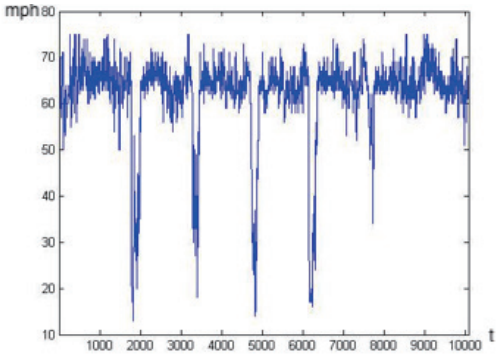


Fig. 1. (Color online) Time series for one road segment.

2.1 Principal component analysis (PCA)

PCA is a popular tool for data analysis by which high-dimensional data are projected onto a low-dimensional subspace while preserving most of the variance in the data. The method is simple and nonparametric.[4] In essence, PCA can be applied to reduce the dimension of a complex data set while revealing the hidden, simplified patterns underlying the data. In the following, $\boldsymbol{x}_t = [x_{1,t}, x_{2,t}, ..., x_{n,t}]^{\mathrm{T}} \in \boldsymbol{R}^n$ is an n-dimensional column vector of average speeds of different road segments at time step t. $\boldsymbol{X}_t = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_t] \in \boldsymbol{R}^{n \times t}$ can be viewed as an $n \times t$ matrix, where a new column is added at each time step t.

There are several ways to explain the PCA technique. One way is to model the vector $\boldsymbol{x}_t$ as a linear combination of k hidden variables. That is, we express $\boldsymbol{x}_t = \boldsymbol{W}\boldsymbol{z}_t$, where $\boldsymbol{z}_t$ are k hidden variables whose values depend on the time step t, and $k \ll n$. $\boldsymbol{W}$ is an $n \times k$ orthonormal matrix to be determined. Since $\boldsymbol{W}$ is orthonormal, $\boldsymbol{W}\boldsymbol{W}^{\mathrm{T}} = \boldsymbol{I}_{k \times k}$. Hence, we deduce that $\boldsymbol{z}_t = \boldsymbol{W}^{\mathrm{T}}\boldsymbol{x}_t$. Using this model, we can reconstruct each $\boldsymbol{x}_t$ using $\tilde{\boldsymbol{x}}_t = \boldsymbol{W}\boldsymbol{W}^{\mathrm{T}}\boldsymbol{x}_t$. Assume that we want to focus on a time window of size T, and that we would like to reconstruct all the data within this window, say, $\boldsymbol{X}_T = [\boldsymbol{x}_1, ..., \boldsymbol{x}_T]$. Then, our optimization (minimizing reconstruction error) can be formulated as

$$\min_{w ortho.} \sum_{\tau=1}^{T} \left\| \boldsymbol{x}_\tau - \left(\boldsymbol{W}\boldsymbol{W}^{\mathrm{T}}\right)\boldsymbol{x}_\tau \right\|^2. \tag{1}$$

Using the singular-value decomposition (SVD) technique, the solution can be expressed as $\boldsymbol{W} = [\boldsymbol{w}_1, ..., \boldsymbol{w}_k]$, where each column $\boldsymbol{w}_i$ is the eigenvector corresponding to the i-th largest eigenvalue of $\boldsymbol{X}_T$. Then, the hidden variables are given by $\boldsymbol{z}_t = [z_{1,t}, ..., z_{k,t}]^{\mathrm{T}} = \left[\boldsymbol{w}_1^{\mathrm{T}}\boldsymbol{x}_t, ..., \boldsymbol{w}_k^{\mathrm{T}}\boldsymbol{x}_t\right]^{\mathrm{T}} \in \boldsymbol{R}^k$. Thus, for any given $k < n$, we can find an orthonormal matrix $\boldsymbol{W}$ and the k hidden variables $\boldsymbol{z}_t$ to reconstruct the data.

2.2 Discovering hidden variables

In this section, we show how to use PCA to find the most important patterns underlying our complex traffic data set. We firstly introduce a conventional PCA method to find the hidden variables so as to reconstruct the data set as accurately as possible. Then, we describe the online method that can update hidden variables at every time step with linear complexity.

Problem Formulation Given n time series corresponding to average speeds on n road segments, updated at each time step t, we aim to determine k hidden variables $\boldsymbol{z}_t$, where $k \ll n$, such that linear combinations of these k hidden variables can be used to reconstruct the data matrix within any time window of size T. Thus, the dimension of the data set is significantly reduced. As a result, we can make more effective, low-cost prediction of speeds in the near future.

2.2.1 Offline PCA pattern discovery

As discussed in Sect. 2.1, we can identify the hidden variables by computing the eigenvectors of the sample covariance matrix of our input data. Then, we can use the first k eigenvectors to reconstruct the data matrix. More details are described in the following algorithm that generates the k hidden variables corresponding to the traffic data of n road segments over a time window of size T.

Algorithm 1. PCA Pattern Discovery

Given: window size T, number of hidden variables k. After receiving every set of T streaming data vectors, we

1. organize the data into an $n \times T$ matrix, i.e., $\boldsymbol{X}_T \in \boldsymbol{R}^{n \times T}$,
2. normalize $\boldsymbol{X}_T$,
3. calculate the k eigenvectors corresponding to the k largest eigenvalues of $\boldsymbol{X}_T$, i.e., $\boldsymbol{w}_1$, ..., $\boldsymbol{w}_k$,
4. compute the k hidden variables $\boldsymbol{z}_t = \left[\boldsymbol{w}_1^{\mathrm{T}} \boldsymbol{x}_t, ..., \boldsymbol{w}_k^{\mathrm{T}} \boldsymbol{x}_t \right]^{\mathrm{T}}$, where $\boldsymbol{x}_t$ is the t-th column of $\boldsymbol{X}_T$, and
5. reconstruct the data matrix $\tilde{\boldsymbol{X}}_T = \boldsymbol{W}\boldsymbol{Z}$.

$\boldsymbol{W} = [\boldsymbol{w}_1, ..., \boldsymbol{w}_k]$ is called the weight matrix. For each element $w_{i,j}$, the magnitude $\left| w_{i,j} \right|$ provides some indication on how much the i-th segment depends on the j-th hidden variable.[4]

2.2.2 Online projection approximation subspace tracking with deflation (PASTd) pattern discovery

The previous PCA algorithm requires the buffering of the data for every time window and a significant amount of computation, namely, computing the first few eigenvectors of the sample covariance matrix, which can be fairly large. The PASTd algorithm, which is based on adaptive filtering techniques and PCA, is an online method that updates the hidden variables and weight matrix incrementally in linear time. The PASTd algorithm has been shown to perform very well in various settings and different applications, such as signal tracking for antenna arrays and image compression.

Algorithm 2. PASTd Pattern Discovery

0. **Initialize**: k orthonormal weight vectors $\boldsymbol{w}_1(0) = [10, \ldots, 0]^{\mathrm{T}} \in \boldsymbol{R}^n$, $\boldsymbol{w}_2(0) = [010, \ldots, 0]^{\mathrm{T}} \in \boldsymbol{R}^n$, etc. $d_i(0)$, $i = 1, \ldots, k$ to a small positive value.
1. As each point $\boldsymbol{x}_t$ arrives, set $\hat{\boldsymbol{x}}_1 = \boldsymbol{x}_t$.
2. For $1 \le i \le k$, we perform the following assignments and updates:

$$\boldsymbol{z}_i(t) = \boldsymbol{w}_i^{\mathrm{T}}(t-1)\hat{\boldsymbol{x}}_i ,$$

$$d_i(t) = \gamma d_i(t-1) + \mathrm{z}_i(t)^2 ,$$

$$\boldsymbol{e}_i(t) = \hat{\boldsymbol{x}}_i - z_i(t)\boldsymbol{w}_i(t-1),$$

$$\boldsymbol{w}_i(t) = \boldsymbol{w}_i(t-1) + \frac{1}{d_i(t)} z_i(t) \boldsymbol{e}_i(t),$$
$$\hat{\boldsymbol{x}}_{i+1} = \hat{\boldsymbol{x}}_i - z_i(t) \boldsymbol{w}_i(t).$$

Algorithm 2 enables the explicit computation of eigencomponents. In fact, $\boldsymbol{w}_i(t)$ is an estimate of the i-th eigenvector at time step t, and $d_i(t)$ is an estimate of the corresponding eigenvalue of the matrix $\boldsymbol{S}_t$, where $\boldsymbol{S}_t = \gamma \boldsymbol{S}_t + \boldsymbol{x}_t \boldsymbol{x}_t^{\mathrm{T}}$. These eigenvalues may be used to estimate the number of hidden variables k, if it is not given. The use of the forgetting factor $0 < \gamma \leq 1$ is intended to ensure that the data matrix $\boldsymbol{S}_t$ is more dependent on the most recent data. Since the traffic data is nonstationary, γ can guarantee the tracking ability and will give more precise estimates of the eigencomponents. The vector $\boldsymbol{e}_i(t)$ is the error between the true data and the reconstruction, and $\boldsymbol{e}_i(t) \perp \boldsymbol{w}_i(t)$. The step for updating the eigenvector $\boldsymbol{w}_i(t)$ can be interpreted as a gradient descent method with a self-tuning step size $d_i(t)$.

In the next section, instead of predicting average speeds, we predict hidden variables. Thus, significant amounts of computational time and energy could be saved. The method of using the PASTd to make travel time prediction is shown as

$$\begin{aligned} \hat{\boldsymbol{z}}(t+1) &= f\left(\boldsymbol{z}(t)\right), \\ \hat{\boldsymbol{x}}(t+1) &= \hat{z}_1(t+1)\boldsymbol{w}_1(t) + \ldots + \hat{z}_k(t+1)\boldsymbol{w}_k(t), \end{aligned} \tag{2}$$

where $f(t)$ is the forecasting hidden variables $\hat{\boldsymbol{z}}(t)$ and $\hat{\boldsymbol{x}}(t+1)$ is the forecasted speed at time t.

3. Short-term Forecasting for Traffic Flow Data

In this section, we start by giving a brief introduction about the k-nearest-neighbor (KNN) method. Then, we show how to apply the KNN method to forecasting the hidden variables over the next brief time horizon. In our work, we found that, for each day of a week, the hidden variables follow similar patterns. Thus, we can forecast the hidden variables for the (l + 1)-th Wednesday by using the hidden variables in the past l Wednesdays.

3.1 KNN method

The KNN method collects historical data as the sample database. In our case, a k-dimensional vector $\boldsymbol{z}_t = [z_{1,t}, \ldots, z_{k,t}]$ is stored, where $z_{i,t}$ is the i-th hidden variable for time step t. Then, the Euclidean distances between all sample points and current data are calculated to generate the KNN's nearest neighbors. Finally, future hidden variables are forecasted by using a weighted average of these KNN's nearest neighbors.

The KNN method is presented in Algorithm 3. To deal with missing data of vehicle speeds, we use the values at the previous time step or the average of the two previous time steps. Thus, we have 1440 time steps for each day. We note that h_f is the forecasted horizon, while h_p is the past horizon.

Algorithm 3. KNN Method

0. **Initialize**: $\boldsymbol{w}$ equal to the values from the last sample data.
1. At time step t during the $(l + 1)$-th week, collect the l historical data $z_{t-hf}(j)$, …, $z_{t-1}(j)$, $1 \le j \le l$.
2. Compute the Euclidean distances between $z_{i,t-hp}(l + 1)$, …, $z_{i,t-1}(l + 1)$ and the l historical data.
3. Find the KNN nearest neighbors with the first KNN shortest distances d_1, …, d_{knn}, and the corresponding weeks are l_1, …, l_{knn}.
4. Forecast the hidden variables with the horizon h_f using $z_{i,t+h_f} = \dfrac{\sum_{j=1}^{knn} \frac{1}{d_j} z_{i,t+h_f,l_j}}{\sum_{j=1}^{knn} \frac{1}{d_j}}$.
5. Forecast the vehicle speeds as $\tilde{\boldsymbol{x}}_{t+h_f} = \boldsymbol{W}\boldsymbol{z}_{t+h_f}$.
6. Update $\boldsymbol{w}$, d using the PASTd method and go to the next time step.

3.2 Forecasting algorithm

Combining the KNN method with the PASTd algorithm, our complete forecasting algorithm is shown in Algorithm 4.

Algorithm 4. Forecasting Algorithm

0. **Initialize**: k, knn, h_p, h_f, l.
1. For each time t, receive the speed $\boldsymbol{x}_t$.
2. Compute the corresponding hidden variables $\boldsymbol{z}(t)$ by the PASTd algorithm.
3. Collect the hidden variables into a matrix for each day for consecutive l weeks as historical data.
4. Forecast the hidden variable for the $(l + 1)$-th week using the KNN method.
5. Forecast $\boldsymbol{x}_{t+h_f}$ through the hidden variables generated at the last step.
6. Compute the error between forecasted and actual speeds.

Updating sample data After the $(l + 1)$-th week, sample data should be updated to forecast the speeds during the $(l + 2)$-th week. In our algorithm, we disregard the sample data in the first week and add the data corresponding to the $(l + 1)$-th week. Thus, the space to store historical data is fixed.

4. Results

4.1 Reconstruction results

In this section, we show the reconstruction results obtained by using both the classical PCA and online PASTd methods. We then compare these two methods in terms of accuracy as well as time efficiency. The data we used are vehicle probe project (VPP) data granted by the RITIS system. In our tests, we chose $n = 48$ road segments over a whole week, which amount to $7 \times$

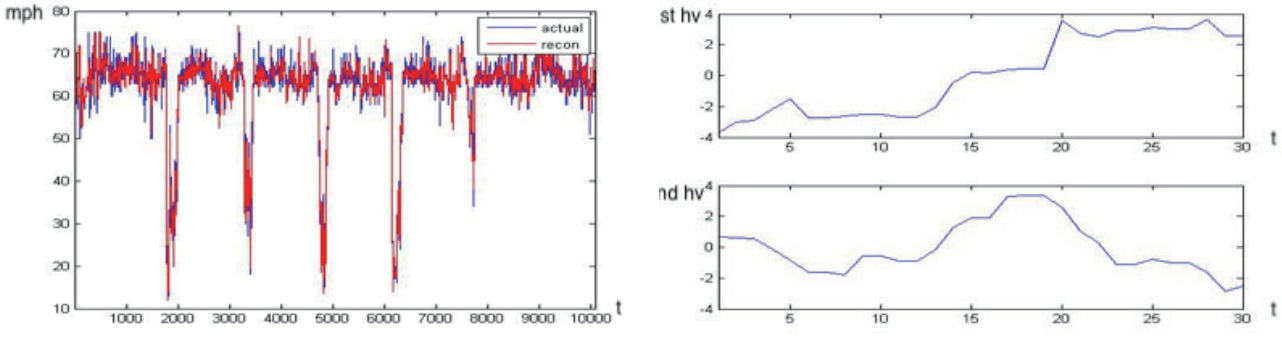
24 × 60 = 10080 vectors each of dimension 48. These 48 segments were randomly chosen from all the road segments of the State of Maryland. The order of the days are Sunday, Monday, ..., to Saturday. If any data is missing, we use the data of the previous time step to fill-in the gap.

4.1.1 PCA performance

In our test, the time window size is selected to be $T = 30$, meaning that we buffer the data and compute the covariance matrix every 30 min. We use $k = 2$ hidden variables to reconstruct the data matrix within each time window.

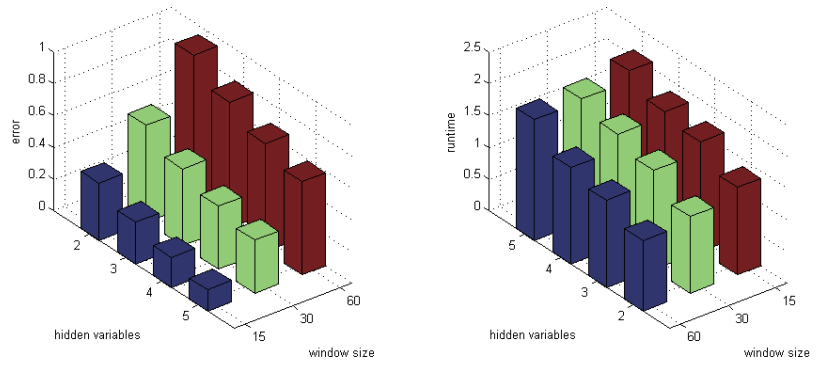
In Fig. 2(a), the blue line shows the original data, while the red line corresponds to the reconstructed data. As we can see, our reconstruction captures the largest statistical variance and has a small deviation from the true data. Figure 2(b) shows the first two hidden variables for one window with size $T = 30$. Note that these are the hidden variables for the normalized data. Thus, the y-axis value illustrates the deviation from the mean. Although this figure represents the patterns for only one time window, we find that the hidden variables for other windows follow similar patterns. Owing to such characteristics, we are able to make predictions for hidden variables.

Figures 3(a) and 3(b) show how changing the values of the parameters affects the performance. In our test, we vary the time window size T from 15 to 60 min and the number of hidden variables from two to five. Here, we use the mean square error (MSE) to value the accuracy and the CPU runtime to value the runtime. In Fig. 3(a), the smaller the window size and the larger the number of hidden variables, the better the performance. In Fig. 3(b), the runtime is in seconds. As we can see, the larger the window size and the smaller the number of hidden variables, the faster the algorithm, which is to be expected. Thus, there is a trade-off between the reconstruction error and the runtime. The results show that the PCA is suitable for traffic flow data reconstruction.



(a) (b)

Fig. 2. (Color online) (a) PCA reconstruction using $k = 2$ patterns for one road segment and (b) hidden variables for one window.



(a) (b)

Fig. 3. (Color online) PCA performance. (a) Reconstruction error and (b) runtime.

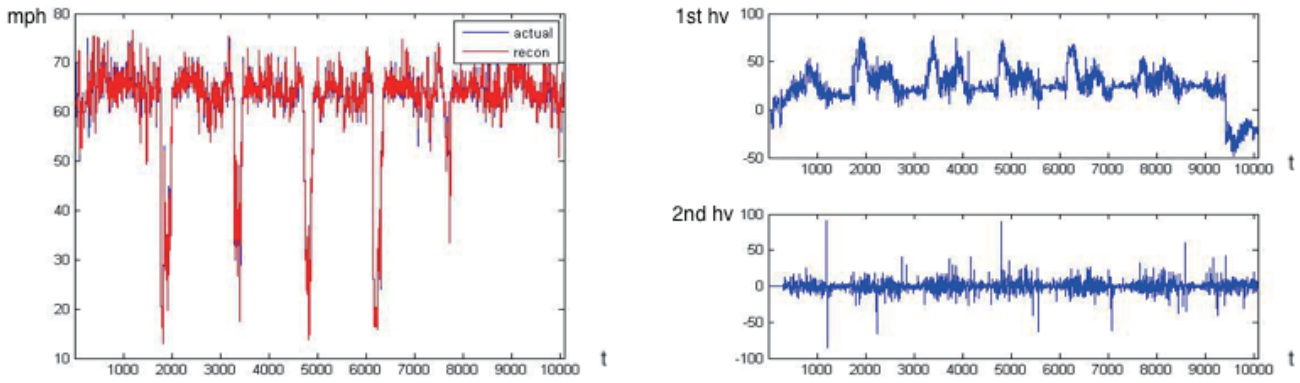
4.1.2 PASTd performance

For PASTd, we use $k = 2$ hidden variables to update the two hidden variables and weight matrix at every time step. In Fig. 4(a), the blue line represents the original data and the red line shows the reconstructed data. We can barely see blue lines in our figures, meaning that our reconstruction is basically the same as the original data. Figure 4(b) illustrates the time series of the two hidden variables determined over a week. As we can see from the time series of the first hidden variable, the five weekdays show similar patterns, while weekend days show different patterns. This can be justified by the fact that weekdays have obvious rush hours, while weekends do not necessarily follow that pattern. Also, the first hidden variable captures the largest variance and hence there will be significant differences between weekdays and weekends.

In PASTd, we update the weight matrix as well as hidden variables incrementally at each time step. The only parameter that matters is the number of hidden variables k. As in the previous subsection, we test the performance of PASTd in terms of reconstruction error and time efficiency as functions of k.

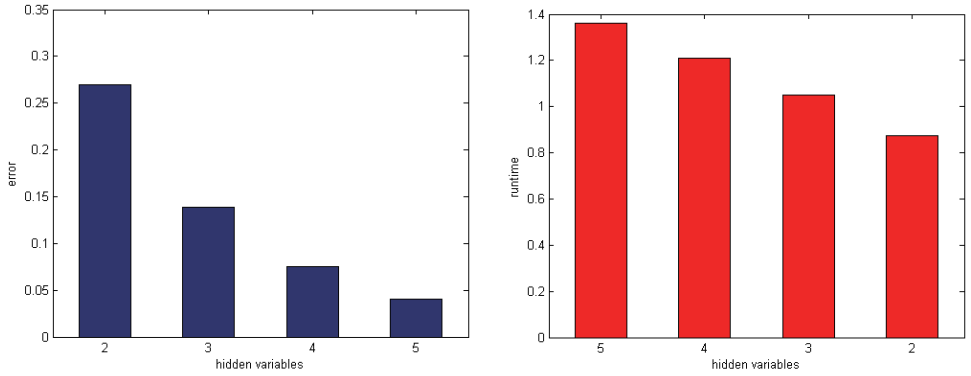
Figures 5(a) and 5(b) show how changing the values of the parameters affects the performance. As we can see, by using a large number of hidden variables, we can obtain a high accuracy, while a small number of hidden variables leads to a short runtime. We note that even for $k = 2$, the error is less than 0.3 mph, while for $k = 5$, the runtime is less than 1.5 s over a week. The results show the high competence and robustness of the PASTd method.

We compare the performance between PCA and PASTd in terms of accuracy and time efficiency. For accuracy, online PASTd outperforms classical PCA probably because PCA captures only the patterns in a single time window, while PASTd incrementally updates the patterns taking into consideration the overall past with more weight assigned to the most recent ones. Another reason might be that PCA is sensitive to the window size T. For time efficiency, PASTd outperforms PCA again because PASTd has a linear complexity $O(n)$, while PCA has at least $O(n^3)$. When n increases, the difference will be substantially larger. Thus, we choose the PASTd method for real-time traffic pattern discovery.



(a) (b)

Fig. 4. (Color online) (a) PASTd reconstruction using $k = 2$ patterns for one road segment and (b) hidden variables over a week.



(a) (b)

Fig. 5. (Color online) PASTd performance. (a) Reconstruction error and (b) runtime.

4.2 Forecasting results

In this section, we show the forecasting results of our proposed algorithm. We choose the MSE and mean absolute proportion error (MAPE) as the performance measurements.

The data used to evaluate the performance are the vehicle speeds collected in Baltimore County, Maryland, which contain 1751 road segments. We set the speeds during the first 40 weeks in 2014 as the sample data and try to forecast the speeds for the 41st week. To obtain the highest time and space efficiency, we choose $k = 1$, $knn = 1$, and $h_p = 1$, and test the performance by varying the forecasting horizon h_f.

Figure 6 shows the forecasting results with $h_f = 60$ for the first road segment, where the blue line represents the actual speeds, while the red line corresponds to the forecasted speeds. As we can see, our forecasting algorithm captures the largest statistical variance and has a small deviation from the true data. Table 1 provides more specific performance characteristics at

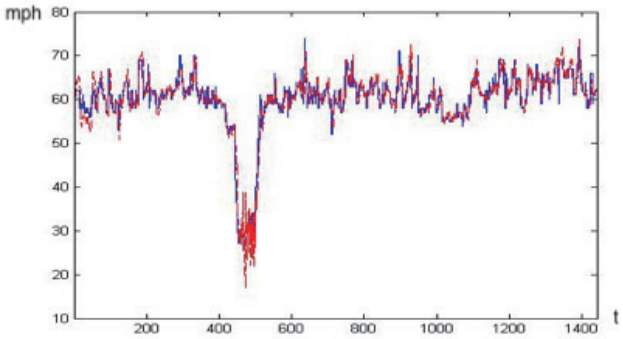
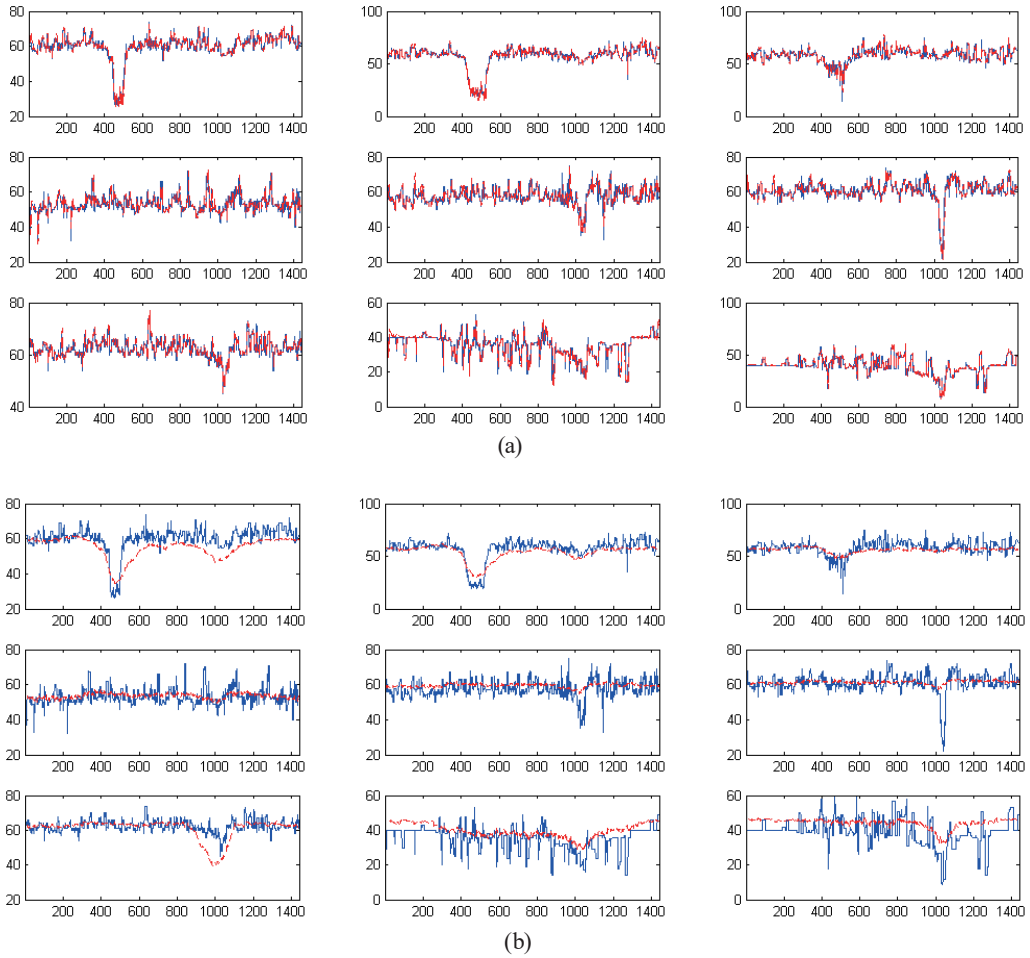


Fig. 6. (Color online) Forecasting results with h_f = 60 for the first road segment.

Table 1
MSE and MAPE with k = 1, knn = 1, and h_p = 1.

h_f	1	5	10	30	60
MSE	5.06	6.01	6.30	7.26	8.74
MAPE (%)	3.75	4.41	4.54	4.83	5.20



(a)

(b)

Fig. 7. (Color online) Forecasting results for the first 9 road segments using (a) our method and (b) historical mean.

various forecasting horizons. When h_f increases, MSE and MAPE do not increase rapidly, verifying the effectiveness and robustness of our forecasting method.

To further show the advantage of our forecasting algorithm, we compare our method with the historical mean method, which also reduces the data dimension to 1. Figure 7(a) shows the forecasting results of our method, while Fig. 7(b) shows those of the historical mean method. As

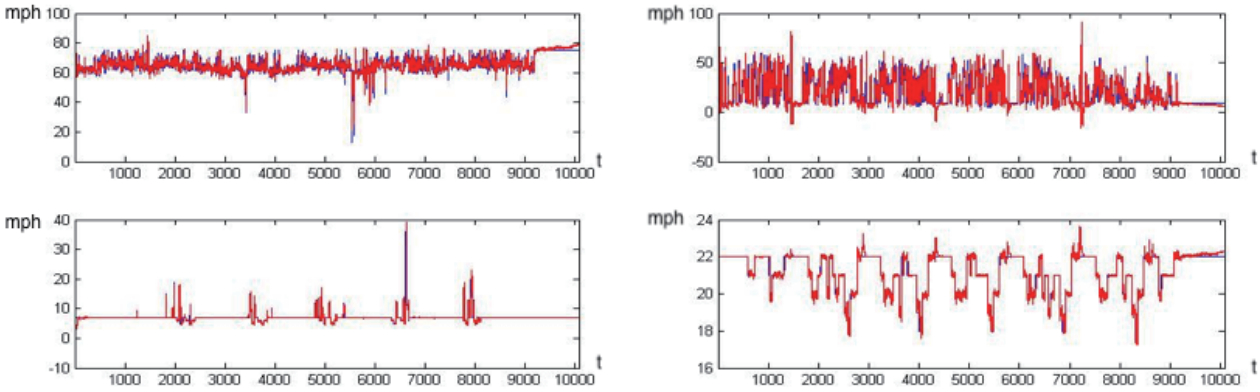


Fig. 8. (Color online) Forecasting results with $k = 1$, $h_p = 1$, and $h_f = 60$, correspond to the worst and best performance characteristics in terms of (a) MSE and (b) MAPE.

we can see, the historical mean method can barely capture the statistical variance. Moreover, its MSE = 45.60, which is much larger than ours.

4.3 Robustness test and outliner detection

The top figure in Fig. 8(a) corresponds to the road segment with the largest MSE, while the bottom figure with the smallest MSE. The road segment whose speed changes frequently may lead to a larger MSE. Note that, even in the worst case, our method can still capture the speed changes and make accurate predictions very quickly. Similarly, the top figure in Fig. 8(b) corresponds to the road segment with the largest MAPE, while the bottom figure with the smallest MAPE. The results are consistent with MSE.

We note that, in both worst cases of MSE and MAPE, the vehicle speeds change too frequently to be realistic. It is reasonable to doubt that the sensors for detecting these two road segments are nonfunctional. As a result, our forecasting algorithm helps detect outliners as well.

5. Conclusions

In this study, we managed to make short-term real-time prediction of traffic flow data of Baltimore, Maryland, US. We applied VPP data from the RITIS system, which are the true world data. PCA is used to derive core traffic patterns from streams of traffic data on a large number of road segments. Furthermore, a more efficient online method, called the PASTd algorithm, is implemented to reduce the data dimension. We use the KNN method to predict the hidden variables. As a result, we are able to forecast the speeds for all the road segments in linear complexity. Our method aims at network-level prediction, regardless of freeway or urban road segments. As far as we know, our method is the first traffic flow data prediction scheme that meets the following requirements: scalability, linear complexity, and no need for data buffering. It also overcomes the restriction of both time and space.

Acknowledgments

This work was supported by the Key Research and Development Program from the Ministry of Science and Technology of China (Grant No. 2018YFB1700200).

References

1 M. S. Ahmed and A. R. Cook: Transp. Res. Rec. **722** (1979) 1. http://onlinepubs.trb.org/Onlinepubs/trr/1979/722/722-001.pdf
2 M. Van Der Voort, M. Dougherty, and S. Watson: Transp. Res. Part C: Emerging Tech. **4** (1996) 307. https://doi.org/10.1016/S0968-090X(97)82903-8
3 I. Okutani and Y. J. Stephanedes: Transp. Res. Part B: Methodol. **18** (1984) 1. https://doi.org/10.1016/0191-2615(84)90002-X
4 S. Papadimitriou, J. Sun, and C. Faloutsos: Proc. 31st Int. Conf. VLDB (2005) 697–708. https://dl.acm.org/citation.cfm?id=1083674
5 Y. Chen, Y. Zhang, and J. Hu: Tsinghua Sci. Technol. **13** (2008) 220. https://doi.org/10.1016/S1007-0214(08)70036-1
6 D. M. Blei, A. Y. Ng, and M. I. Jordan: J. Mach. Learn. Res. **3** (2003) 993. http://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf
7 X. Wei, J. Sun, and X. Wang: IJCAI **7** (2007) 2909. http://www.aaai.org/Papers/IJCAI/2007/IJCAI07-468.pdf
8 C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu: Proc. 29th Int. Conf. VLDB **29** (2003) 81. https://pdfs.semanticscholar.org/a4bb/55eeefbf8f50d932adeb10699fd0bef92cc5.pdf
9 Y. Sakurai, S. Papadimitriou, and C. Faloutsos: Proc. 2005 ACM SIGMOD Int. Conf. Management of Data (2005) 599–610. https://dl.acm.org/citation.cfm?id=1066226
10 Y. Zhu and D. Shasha: Proc. 28th Int. Conf. VLDB (2002) 358–369. https://doi.org/10.1016/B978-155860869-6/50039-1
11 P. Cai, Y. Wang, G. Lu, P. Chen, C. Ding, and J. Sun: Transp. Res. Part C: Emerging Technol. **62** (2016) 21. https://doi.org/10.1016/j.trc.2015.11.002
12 B. M. Williams and L. A. Hoel: J. Transp. Eng. **129** (2003) 664. https://doi.org/10.1061/(ASCE)0733-947X(2003)129:6(664)
13 S. Lee and D. Fambro: Transp. Res. Rec. (1999) 179–188. https://doi.org/10.3141/1678-22
14 J. Wang, W. Deng, and Y. Guo: Transp. Res. Part C: Emerging Technol. **43** (2014) 79. https://doi.org/10.1016/j.trc.2014.02.005
15 J. Guo, W. Huang, and B. M. Williams: Transp. Res. Part C: Emerging Technol. **43** (2014) 50. https://doi.org/10.1016/j.trc.2014.02.006
16 Y. Hu, C. Wu, and H. Liu: Transport **26** (2011) 197. https://doi.org/10.3846/16484142.2011.593121
17 X. Jiang and H. Adeli: J. Transp. Eng. **131** (2005) 771. https://doi.org/10.1061/(ASCE)0733-947X(2005)131:10(771)
18 B. L. Smith and M. J. Demetsky: Transp. Res. Rec. (1994) 1453. https://doi.org/10.1109/ICSMC.1994.400094
19 S. Sun, R. Huang, and Y. Gao: J. Transp. Eng. **138** (2012) 1358. https://doi.org/10.1061/(ASCE)TE.1943-5436.0000435
20 A. Stathopoulos, L. Dimitriou, and T. Tsekeris: Comput.-Aided Civ. Infrastruct. Eng. **23** (2008) 521. https://doi.org/10.1111/j.1467-8667.2008.00558.x
21 J. Van Lint and C. Van Hinsbergen: Artif. Intell. Appl. Crit. Transp. Issues **22** (2012) 22. http://onlinepubs.trb.org/onlinepubs/circulars/ec168.pdf
22 E. I. Vlahogianni, M. G. Karlaftis, and J. C. Golias: Transp. Res. Part C: Emerging Technol. **43** (2014) 3. https://doi.org/10.1016/j.trc.2014.01.005